CS F213: OOP Lab 6
Interfaces & Wrappers

Avi Wadhwa, Jaskaran Singh Bhatia, Sarvesh Borkar

30th March 2021

The objective of this lab is to test your understanding on the concept of interfaces & wrappers. In this lab, you will be designing a database for a marketplace and simulate transactions in marketplace using this database model.

General Instructions -

Read the following instructions very carefully before starting your work -

1. Make sure that the source code is written in the default package.
2. Ensure that the name of each file is the same as the corresponding class name.
3. Ensure that the name of methods is the same as those given in Javadoc. Failure to comply may cause deduction in credit even if the implementation is correct.
4. Do not copy directly the names of methods from Javadoc. Write them out yourselves.
5. A template file will be provided for **Market** class. Do not modify existing methods in the template.
6. You are required to upload all classes/interfaces at once. It is not possible to test individual classes on codepost.

7. A brief description of the classes and methods to be used is given further in the problem statement but you must refer to the Javadoc for the complete description.
8. Be sure to check your code on the test cases provided locally (as jar file), before submitting on Codepost. Remember that Codepost takes only the latest submission and not the best submission into consideration.
9. The suggested order of writing the code:
    a) Record
    b) User
    c) Product
    d) Transaction
    e) IDatabase
    f) IEditableDatabase
    g) Database
    h) UserDatabase
    i) ProductDatabase
    j) TransactionDatabase
    k) Market

Overview:

- Each Database consists of a set of records.
- Records can be of three types, **User**, **Product** and **Transaction**
- Each **Product** is owned by a **User.**
- In a **Transaction**, a **Product** is sold by a **User**(seller) and bought by another **User**(buyer).
- Each of these three Records are stored in **UserDatabase**, **ProductDatabase** and **TransactionDatabase** respectively.

- Databases can be implemented using two interfaces, ***IDatabase*** and ***IEditableDatabase.***
- Databases implementing ***IDatabase*** can insert and retrieve records.
- Databases implementing ***IEditableDatabase*** can insert, retrieve, edit and delete records.
- **TransactionDatabase** supports only insertion and retrieval while **UserDatabase** & **ProductDatabase** supports all four operations, insertion, retrieval, editing and deletion.
- **Market** has three databases, **UserDatabase**, **ProductDatabase** and **TransactionDatabase.**
- In **Market**, we can perform the following operations:
    - Add product and user records to their respective databases.
    - Verify balance of a user.
    - Verify ownership of a product**.**
    - Carry out transactions between users.
- A transaction for a product can be carried out between two users only if seller owns the product & buyer has sufficient balance to buy it.

Note: It is recommend to implement the getTransactedValue() function in the TransactionDatabase class after the rest of the code has been implemented.

Class Descriptions:

1. Record

- **Fields**
    - id
- **Constructor**
    - Record(id)

- **Methods**
  - getId()
  - changeId()

2. User

- **Fields**
  - name
  - balance
- **Constructor**
  - User(id, name, balance)
- **Methods**
  - getName()
  - getBalance()
  - updateBalance()

3. Product

- **Fields**
  - name
  - cost
  - owner
- **Constructor**
  - Product(id, name, owner, cost)
- **Methods**
  - getName()
  - getCost()
  - getOwner()
  - changeOwner()

4. **Transaction**
    - **Fields**
        - buyer
        - seller
        - product
    - **Constructors**
        - Transaction(id, buyer, seller, product)
    - **Methods**
        - getBuyer()
        - getSeller()
        - getCost()

5. **IDatabase (**Interface**)**
    - **Methods**
        - getNumRecords()
        - getRecord()
        - putRecord()

6. **IEditableDatabase (**Interface**)**
    - **Methods**
        - deleteRecord()
        - editRecord()

7. **Database**
    - **Fields**
        - records
        - numRecords
        - MAX_RECORDS
    - **Constructors**
        - Database()
    - **Methods**
        - getNumRecords()
        - getRecord()
        - putRecord()

8. **UserDatabase**
    - **Constructors**
        - UserDatabase()
    - **Methods**
        - deleteRecord()
        - editRecord()

9. **ProductDatabase**
    - **Constructors**
        - ProductDatabase()
    - **Methods**
        - deleteRecord()
        - editRecord()

10. **TransactionDatabase**
    - **Constructors**
        - TransactionDatabase()
    - **Methods**
        - getTransactedValue()

11. **Market**
    - **Fields**
        - userDatabase
        - productDatabase
        - transactionDatabase
    - **Constructor**
        - Market()
    - **Methods**
        - addUserRecord()
        - addProductRecord()
        - removerFromSeller()
        - addToBuyer()
        - verifyOwnership()
        - verifyBalance()
        - transaction()

Test Cases:

Methods	Marks allotted
getName()	1
getBalance()	1
changeOwner()	1
editRecord()	1
verifyOwnership()	1
verifyBalance()	1
addToBuyer()	1
removeFromSeller()	1
transaction()	2
getTransactedValue()	2
Total	12

Note: The score will be scaled down to same weightage as previous labs (7 mark)