Ceng351 - Data Management And File Structures

Programming Assignment 1

Tauros AVM

Due Date: November 18, 2012

1 Story

In this homework you will help IT Department of Tauros AVM (AVM is a shortcut for 'shopping mall' in Turkish) to develop their database system. Tauros AVM is a shopping mall which is so important for our country that it worths cutting thousands of trees to be able to build a road passing near it. So, we should join our forces to build the best database system ever. Using this system the IT Department will be able to manage stores, loyalty card users, their shoppings and special offers to the users.

Members of IT Department of Tauros AVM are good at graphical design and front pages but they don't have a deep background on relational database systems. Moreover, there is no enough time to study and become an expert on database systems because the build of the road is almost done, thanks to the mayor and the police department for their great effort! Therefore they asked me to design and implement all database related parts and I answered that METU students would be happy to help this great shopping mall to operate better. We owe this to them because of their high contribution to bring civilization to METU.

Because they will design and implement the front pages, they should communicate the database we will design. They prepared some interfaces to specify the functionality of the database they need. Your goal is to implement these interfaces to let them communicate with the database without knowing the details of the database design.

2 Objectives

Upon completion of this homework you will be familiar with the following concepts:

- Java programming language basics
- Object oriented programming perspective
- Connecting and querying to MySQL Server using JDBC
- Designing a database for a specific task

3 Tasks

3.1 Designing Database

You are asked to design a database for a shopping mall which can accomplish following tasks:

- Adding / removing stores
- Get / set store information
- Register / unregister users
- Keep track of shoppings
- Search for stores, user and shoppings
- Managing offers

Necessary data for you to accomplish these tasks will be provided. All of these tasks are explained below in more detail. Interfaces and enumerations you will use to accomplish these tasks will be provided to you and explained in *Reference* part.

Adding / Removing Stores
You will be provided name and category of the store to add it to the database. Categories classify stores according to items sold inside. A store can have only one category. There might be two store with the same name but different category. There won't be stores with the exactly same name and category. Therefore, you should raise an exception if an existing store is tried to be added to the database.

For removing a store, you will be provided the store object. If a nonexisting store is tried to be deleted you should raise an exception.

Getting / Setting Store Information
Store information is editable. But, you should check if a change in the store information creates a duplicate. If there exist such a store before you should raise an exception.

Registering / Unregistering Users
Users will be registered to the system using their full name (name and surname), id numbers (11 digits) and adresses (max. 500 characters). There might be users with the same name. Id numbers will be unique. If an existing user is tried to be registered again you should raise an exception.
Users will be unregistered using user objects. If a nonexisting user is tried to be deleted you should raise an exception.
Keeping Track of Shoppings
User's shoppings will be stored in the database. Shoppings will be added to the database using user info, store info, date and the amount of the shopping. Shopping information will be also used for special offers for the users.

Searching for Stores, Users and Shoppings

Stores can be searched using only name, only category or name and category. When searched with only name, stores with name which includes the search pattern exactly will be returned. For example, when searched with pattern 'ark', 'Arkadas' and 'Park Bravo' will be returned but 'Darty' will not. Same things happen when searched with name and category but, this time category of the returned stores should also match the searched category. When searched using only category, all stores belonging that category should be returned. For all cases, if more than one result will be returned they should be sorted alphabetically according to store names.

Users can only be searched using their full names. You will use the same strategy with stores. For example, if search pattern is 'Emr', users with name 'Emre Isikligil' and 'Emrah Yildiz' will be returned but, 'Emir Kaplan' will not. When more than one user matched they should be sorted alphabetically according to their name.

Shoppings will be searched using user information, date interval and store information (optional). It will also be possible to get all shoppings of a user and all sales of a store for a specific date. Shopping information should returned as sorted from new to old.

Managing Offers

There are three types of special offers which are 'Prime Minister Offer', 'Governor Offer' and 'Mayor Offer'. You should be able to check if a user has right to claim a specific offer. Details of the offers are described below.

Prime Minister Offer offers you a job in any ministry. You can use this offer for your close relatives. To be able to claim this offer a user should have been spent 100 TL or more in five (or more) different stores in the last month.

Governor Offer offers you an id card which makes you untouchable against police officers for six months. To be able to claim this offer a user should have been spent 50 TL (or more) in the same store five times (or more) in different days in the last month.

Mayor Offer offers you subscription to a communication group in which plans of new roads to be built by the Municipality of Ankara are shared. Hence, a user who claim this offer can buy cheap properties near new roads and sell them much more expensive after the roads are built. To be able to claim this offer a user should shop at least 10 times in different days in the last month (Amount is unnecessary).

3.2 Creating Database

When I execute your homeworks for the first time the database will be totally empty. So, you are supposed to implement some methods to create everything needed to accomplish tasks mentioned above. In other words, you will create your databases programatically when called. Each of you will

be provided a user account which has necessary controls over a database. When you are asked to do so you should create all tables, views, triggers, etc if they are not created before. If they are created before you should raise an exception.

3.3 Implementing Interfaces

Because members of IT Department of Tauros AVM do not know much about Relational Database Management Systems, they don't want to make their hands dirty by getting down into details of database operations. Therefore they have defined some interfaces which provide isolation between the data they need and the database operations. You should create classes which implement these interfaces. So, they can manipulate all data by calling the methods defined in the interfaces.

You have four interfaces to implement which are IShoppingMall, IUser, IStore and IShopping. Reference to the interfaces are in *Reference* part.

3.4 Editing Starter Class

Starter class has only one static method which returns the handler of the IshoppingMall object. You should edit the commented statement of *getShoppingMallHandler* function by changing "*YourShoppingMallClass()*" with the constructor of the class defined by you which implements *IShoppingMall* interface. After you should remove the comment tag.

For example, if the name of the class which implements *IShoppingMall* interface is My*ShoppingMall*, edited version of *getShoppingMallHandler* function will contain following statement:

```
shopping_mall = (IShoppingMall)new MyShoppingMall();
```

4 Reference

```
public interface
IShoppingMall
```

Entry point for the application. When the application is started an instance of IShoppingMall will be created using *getShoppingMallHandler* function and all other operations will be done through it.

Methods

```
public void onStart()
```

Called when an instance of the implementing class is created. Any initialization may be placed inside this function. Do not call it manually.

```
public boolean createDatabase()
```
Creates necesssary database elements (tables, triggers, views) unless they have been created before.

Returns

True if successful, false otherwise.

Throws

DatabaseAlreadyCreated exception if database is created before.

```
public void addStore(String storeName, Category category)
```
Adds the specified store to the database unless it has been added before.

Parameters

storeName	name of the store to be added
category	category of the store to be added

Throws

StoreAlreadyExists exception if the store has been added before

```
public void removeStore(IStore store)
```
Removes the specified store from the database if it exists.

Parameters

store	instant of the IStore object to be removed

Throws

StoreNotExist exception if the store does not exist.

```
public void registerUser(String fullname, String id, String address)
```
Registers the specified user by adding it into the database unless it has been added before.

Parameters

fullname	fullname of the user to be registered
id	11-digit id number of the user to be registered
adress	address of the user to be registered (max 500 character)

Throws

UserAlreadyExists exception if the user has been added before

```
public void unregisterUser(IUser user)
```
Unregister the specified user by removing it from the database if it exists.

Parameters

user	instant of the IUser object to be unregistered

Throws

UserNotExist exception if the user does not exists.

```
public IStore[] searchStore(String nameIncludes)
```

Searches for the stores name of which includes the given pattern.

Parameters

nameIncludes pattern to be used for searching stores

Returns

Array of IStore objects whose names include the given pattern in alphabetical order. Returns null if there is no matching store.

```
public IStore[] searchStore(String nameIncludes, Category category)
```

Searches for the stores name of which includes the given pattern and category of which equals to the given category.

Parameters

nameIncludes pattern to be used for searching stores

category Category enumerataion to be used for searching stores

Returns

Array of IStore objects whose names include the given pattern and whose categories are equal to the given category in alphabetical order. Returns null if there is no matching store.

```
public IStore[] searchStore(Category category)
```

Searches for the stores category of which equals to the given category.

Parameters

category Category enumerataion to be used for searching stores

Returns

Array of IStore objects whose categories are equal to the given category in alphabetical order. Returns null if there is no matching store.

```
public IUser[] searchUser(String nameIncludes)
```

Searches for the users whose full names include the given pattern.

Parameters

nameIncludes pattern to be used for searching users

Returns

Array of IUser objects whose names include the given pattern in alphabetical order. Returns null if there is no matching user.

```
public IUser getUser(String id)
```

Gets the user whose id is equal to the given id.

Parameters

id id of the user to be returned

Returns

IUser object for the user whose id is equal to the given id. Returns null if there is no matching user

```
public interface
IStore
```

Designed to carry out store related operations.

```
public String getStoreName()
```

Returns

The name of this store

```
public Category getStoreCategory()
```

Returns

The category of this store as Category object

```
public void setStoreName(String newName)
```

Changes name of this store unless there exists a store with the same name and the category.

Parameters

newName the name to be assigned as the new name of this store

Throws

StoreAlreadyExists exception if there exists a store with the same name and the category.

```
public void setStoreCategory(Category newCategory)
```

Changes category of this store unless there exists a store with the same name and the category.

Parameters

newCategory the category to be assigned as the new category of this store

Throws

StoreAlreadyExists exception if there exists a store with the same name and the category.

```
public IShopping[] getShoppings(String startDate, String endDate)
```

Returns the list of shoppings made in this store in the given date interval.

Parameters

startDate	Starting date of the interval (included to the interval)
endDate	End date of the interval (included to the interval)

Returns

Array of IShopping objects for the returned shoppings in the order from new to old.

```
public IShopping[] getShoppings(String date)
```

Returns the list of shoppings made in this store in the given date.

Parameters

date	date in which the shoppings made in this store to be returned

Returns

Array of IShopping objects for the returned shoppings in the order from new to old.

```
public IShopping[] getShoppings()
```

Returns the list of all shoppings made in this store.

Returns

Array of IShopping objects for the returned shoppings in the order from new to old.

```
Public interface
```

IUser

Designed to carry out user related operations.

```
public String getFullName()
```

Returns

The full name of this user.

```
public String getId()
```

Returns

11-digit id number of this user.

```
public String getAddress()
```

Returns

The address of this user.

```
public void updateAddress(String newAdress)
```

Updates the address of this user

Parameters

newAddress new address of this user (max 500 characters)

```
public void shopped(IStore store, String date, double amount)
```

Called when this user shops. Shopping with the given information should be stored when this function is called.

Parameters

store the store where this user shopped in
date date of the shopping
amount amount of the shopping

```
public IShopping[] getShoppings(IStore store, String startDate,
                                String endDate)
```

Returns the list of shoppings of this user made in the given store in the date interval.

Parameters

store the given store
startDate starting date of the interval (included to the interval)
endDate end date of the interval (included to the interval)

Returns

Array of IShopping objects for the returned shoppings in the order from new to old.

```
public IShopping[] getShoppings(IStore store)
```

Returns the list of all of the shoppings of this user made in the given store.

Parameters

store the given store

Returns

Array of IShopping objects for the returned shoppings in the order from new to old.

```
public IShopping[] getShoppings()
```

Returns the list of all shoppings of this user.

Returns

Array of IShopping objects for all of the shoppings of this user in the order from new to old.

```
public boolean checkPrimeMinisterOffer()
```

Checks if this user has right to claim Prime Minister Offer.

Returns

true if this user has right to claim the offer, *false* otherwise.

```
public boolean checkGovernorOffer()
```

Checks if this user has right to claim Governor Offer

Returns

true if this user has right to claim the offer, *false* otherwise.

```
public boolean checkMayorOffer()
```

Checks if this user has right to claim Mayor Offer

Returns

true if this user has right to claim the offer, *false* otherwise.

```
public interface
IShopping
```

Designed to carry out shopping related operations

```
public IUser getUser()
```

Returns

The IUser object for the user who did this shopping.

```
public IStore getStore()
```

Returns

The IStore object for the store in which this shopping is done.

```
public String getDate()
```

Returns

The date of this shopping.

```
public double getAmount()
```

Returns

The amount of this shopping.

6 Specifications

- Homework must be implemented in Java.
- You must use JDBC for the database operations.
- You will be provided a user account on MySQL Server on my office machine. You can also install MySQL Server on your local machines to work on a local database. However final version of your homework must carry out all database operations on the account on my office machine. So make sure that the connection information is the right one before submitting it.
- Changes in any object (name, category, ...) should be reflected to the database immediately.
- All text based inputs will be in ASCII characters.
- Format of the all date inputs and outputs will be "DD.MM.YYYY".
- When sorting outputs if two or more objects have the same value you do not need sort them between each other. But, they should be ordered consecutively.
- All of the classes should be defined in "com.tauros" package.
- You cannot change interface and class definitions which I provide to you.
- You won't submit interface and class definitions. I will add them to your project. You can use ones I provide to you while developing the homework.
- You won't implement the main function. I will implement it to test your homework. You will submit only your class definitions and the edited Starter Class file.
- Your homeworks will be evaluated on Inek machines (64-bit). So make sure that your homework works on Inek machines.
- Everything you submit must be your own work. Do not copy any piece of code from Internet or your friends. Both source and destination will be counted as cheated.
- You must follow the course page on the newsgroup requlary. Any change and clarification will be announced on the newsgroup.
- You can ask any question about homework or anything related on newsgroup.

7 Submission

- You will submit all class definiton files and edited version of *Starter.java* in a archive file named "hw1.tar.gz".

- Your files should be in the root folder. I mean do not put your files in a folder and create archive of that folder. You can simply call following command on linux machines:

```
tar cvf hw1.tar.gz Starter.java file1.java file2.java ...
```

- Submission will be done through cow.

8 Useful Links

- **SQL Statement Syntax**
  http://dev.mysql.com/doc/refman/5.5/en/sql-syntax.html
- **MySQL Tutorial**
  http://dev.mysql.com/doc/refman/5.5/en/tutorial.html
- **JDBC Basics**
  http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html
- **Java Language Basics**
  http://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html
- **Eclipse For Java Developers (Recommended)**
  http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/junosr1

Kolay gelsin.
Emre