CS310 Final Project

Goal: Draft a basic inventory system for a concession stand.

The Final Project Due Date and time will be confirmed by the instructor.
Late deliverables will result in 2.5% loss of credit *per day* - as stated in the syllabus. Final Project is worth 30% of your final grade. (This is estimated to take 8-10 hours.)

Overview:

- Create 3 tables. They will meet certain specifications defined in the requirements.
- Create CRUD (Create, Read, Update, Delete) procedures to meet specific use cases.
- Create a procedure to report on the data, as defined in the requirements below.
- Create java code to interact with the database, i.e. to call the procedures in the database.

Technical Requirements

Start with Database Structures to meet the following requirements

1. Provide a table for concession items. `Item`
    a. Columns
        i. ID int - ID will auto increment and be a primary key
        ii. ItemCode varchar(10) - a required value. ItemCode cannot be duplicated
        iii. ItemDescription varchar(50)
        iv. Price decimal(4,2) - Price must Default to 0 if no data is provided
2. Provide a storage table for purchases. `Purchase`
    a. Columns
        i. ID int - ID will auto increment and be a primary key
        ii. ItemID int - This will be a Foreign Key to Item.ID, it will be a required value
        iii. Quantity int - Quantity will be a required value
        iv. PurchaseDate datetime - PurchaseDate will default to “today at this time” when no input is provided
3. Provide a storage table for `Shipment`
    a. Columns
        i. ID int - ID will auto increment and be a primary key
        ii. ItemID int - This will be a Foreign Key to Item.ID, it will be a required value
        iii. Quantity int - Quantity will be required
        iv. ShipmentDate date - ShipmentDate will be a required value and unique (only one per day allowed)

Then Create a java [command line] application to meet the following requirements

Optional: Appropriate procedures are useful SQL structures to support your java application in running database code and to make the Java code more concise and potentially safer from SQL injection attacks. Using procedures, you can get started on the Final Project even before you’ve learned the Java code component. **Note**: Parameters in procedures are best when they are NOT named the same as table columns, errors in results may arise if you do not follow this practice.

Notes:

- Commands below are defined [approximately] in order of what you can start on - based on class learning - not necessarily in order of use cases.
- In using command line arguments - order is crucial to testing. *Failure to meet the command line arguments order will result in failed tests and lower grade.*
- The class must be named "project" for tests to work as expected. Failure to name your class "project" will result in failed tests and lower grade.
- Some errors will arise through my testing, this may be expected or unexpected by the test. There are some natural limitations caused by the constraints on the tables in the database. Allow natural errors to be visible in the output, be careful when deciding if an error is appropriate or inappropriate. If an error results from the constraints on the tables, allow them to show in the command line output. (For example, I should not be able to delete and Item that is referenced by a Shipment, that causes a foreign key violation.)
- As a developer of data applications, you may note that there are concerns, inherent limitations and potential for issues about this design, that's understood. This is a simplified set of requirements to test your basic skills. This design would be incomplete for a production-quality application.

I will compile using this command: `javac FinalProject.java`

Allow these Command Line Requests and parameters:
Each numbered item is worth 5 points except ItemsAvailable, which has 3 specific test cases (see the hints on that request) and is worth 15 points on it's own. Partial credit is possible, but generally they work or they don't.
There are also 5 points that I will use to subtract for difficult to use submissions and unexpected errors - if additional work is required based on imprecise submissions, I will use these points to deduct score. (i.e. if you require me to send Item.ID instead of ItemCode, it will reduce your "ease of use" points, but I will do my best to test anyway.)

1. `java project` /?
    a. "Usage" text will be output for the "/?" command line argument
    b. Provide an output for each of the following requests.
    c. "project " is the name of the class. "FinalProject" is the name of the file.
2. `java project` CreateItem <itemCode> <itemDescription> <price>
    a. Creates an item
3. `java project` CreatePurchase <itemCode> <PurchaseQuantity>
    a. Creates a Purchase record
4. `java project` CreateShipment <itemCode> <ShipmentQuantity> <shipmentDate>
    a. Creates a Shipment record
5. `java project` GetItems <itemCode>
    a. Allow % as input to be used for all Items
6. `java project` GetShipments <itemCode>
    a. Returns shipments associated with the itemcode
    b. Allow % as input to be used for all Items
7. `java project` GetPurchases <itemCode>
    a. Returns purchases associated with the itemcode
    b. Allow % to be used for all Items
8. `java project` ItemsAvailable <itemCode>
    a. This is the most complex requirement. Test it carefully.
    b. Returns a *calculation* for items. Simply stated, it will return, per item requested: all shipment quantities *minus* all purchase quantities.
    c. Only one record in the result set per item code.
    d. Return a minimum of these columns: ItemCode, ItemDescription, and the number of items available.
    e. Allow % to be used for a request, this will return all Items
    f. Hints and test cases:
        i. Your result **must** show 0 for an existing item, *even if no shipments or purchases exist.*
        ii. Your result must show accurate quantities, *even if no purchases exist.*

iii. Your result must show accurate quantities, *even if many shipments and/or many purchases exist for the item.*
Just for fun: https://tmblr.co/Z14uHt2aAQXc7

9. `Java project` UpdateItem <itemCode> <price>
    a. Changes the price for the itemItemCode
10. `java project` DeleteItem <itemCode>
    a. Removes only the item exactly matching the parameter (errors are expected if shipments or purchases are referencing the item.)
    b. Do not delete shipments or purchases. *Some errors are expected to transmit to the command line if I try to delete an item that still has purchases or shipments.*
11. `java project` DeleteShipment <itemCode>
    a. Removes only the most recent shipment for one item code, if a shipment exists.
12. `java project` DeletePurchase <itemCode>
    a. Removes only the most recent purchase for one item code, if a purchase exists.

Submission Instructions:

This submission command assumes that the java code is saved in a folder named "project":
cd project

```
submit nikeshjoshi cs310 cs310_2019_001_project
```

- Failure to submit into the appropriate folder could result in no credit for submitting.
- Note: Tests will run using your database and using your port and your password. You may wish to change your password since it must appear in your java file. In MySQL you can change your password by using the following command. If I cannot run the java due to mismatched passwords, all tests will fail and you (and your partner) will receive no grade. Test your code before submitting.
    - `SET PASSWORD FOR 'root'@'localhost' = PASSWORD('MyNewPasswordHere');`
- Write code to meet the requirements.
    - Save 1 (or 2) files of java code into a project folder in onyx.

Some Example unit Tests are included below, testing them will improve your code and grade, expand the unit tests to check every requirement to achieve the best grade possible. The grader will be using a very long list of tests.
    - Simple SQL Injection attacks are valid tests for graders. You will want to protect your database against obvious SQL injection attacks.

This project will exercise and test your proficiency in these areas:

- Creating tables with constraints.
- CRUD (Create, Read, Update, Delete) operations on database tables.
- Using SQL parameters.
- Using variables and/or subqueries.
- Using Predicates.
- Joining Tables.
- Using Aggregate Functions and GROUP BY.
- Using a java application to interact with a database.
- Running unit tests.

Sample Unit Tests

//The following show you some unit tests to get you started. Argument order is important

// the following will show “Usage” options in the output.
```
java project /?
```

// the following will create an item
```
java project CreateItem LSoda "Large Soda" 5.50
```

// the following will not create a second item of the same code as the previous - an error will be returned
```
java project CreateItem LSoda "Large Soda" 5.50
```

// the following will return all items
```
java project GetItems %
```

// the following will return one item
```
java project GetItems LSoda
```

// the following will update only one item to have a new price
```
java project UpdateItem LSoda 6.00
```

// if tests are prepared in the order provided, 0 available items will show for LSoda
```
java project ItemsAvailable %
```

```
java project CreateShipment LSoda 15
```

// if tests are prepared in the order provided, 15 available items will show for LSoda
```
java project ItemsAvailable %
```

```
// continue on your own
```